Date Extraction and Translation using Neural Machine Translation

Venkateshwarlu Vennnam
Department of Information Technology, GITAM University

Dr.Riyazuddin.Y.Md
Assistant Professor- Dept of IT, GITAM,Hyderabad

***Abstract*— This Neural Machine Translation (NMT) model to translate human readable dates ("16th of December, 2012") into machine readable dates ("2012-12-16"). Machine learning model is built using an Attention model, one of the most sophisticated sequence to sequence models and deep learning neural networks. The model accepts a date as input written in a variety of possible formats (e.g. "the 29th of August 1948", "03/29/1948", "29 JUNE 1987") and translate them into standardized, machine readable dates (e.g. "1948-08-29", "1948-03-29","1987-06-29"). The output format would be the common machine-readable format YYYY-MM-DD. This model can also be scaled up to translate from one language to another, such as translating from English to Hindi but language translation requires massive datasets and usually takes days of training on GPUs.**

Keywords*— *Dates, sentences, encode, decode, sequence models, RNN, attention, LSTM, Deep Learning, Neural Networks

I. INTRODUCTION

Machine translation is the task of automatically converting source text in one language to text in another language.

In a machine translation task, the input already consists of a sequence of symbols in some language, and the computer program must convert this into a sequence of symbols in another language. Given a sequence of text in a source language, there is no one single best translation of that text to another language. This is because of the natural ambiguity and flexibility of human language. This makes the challenge of automatic machine translation difficult, perhaps one of the most difficult in artificial intelligence.

The fact is that accurate translation requires background knowledge in order to resolve ambiguity and establish the content of the sentence.

Classical machine translation methods often involve rules for converting text in the source language to the target language. The rules are often developed by linguists and may operate at the lexical, syntactic, or semantic level. This focus on rules gives the name to this area of study: Rule-based Machine Translation, or RBMT.

RBMT is characterized with the explicit use and manual creation of linguistically informed rules and representations.

The key limitations of the classical machine translation approaches are both the expertise required to develop the rules, and the vast number of rules and exceptions required.

II. PROPOSED APPROACH

A. Statistical Machine Translation

Statistical machine translation, or SMT for short, is the use of statistical models that learn to translate text from a source language to a target language gives a large corpus of examples.

Statistical model as follows:

Given a sentence T in the target language, I seek the sentence S from which the translator produced T. We know that our chance of error is minimized by choosing that sentence S that is most probable given T. Thus, I wish to choose S so as to maximize Pr(S|T).

B. Neural Machine Translation (NMT):

Neural machine translation, or NMT for short, is the use of neural network models to learn a statistical model for machine translation.

The key benefit to the approach is that a single system can be trained directly on source and target text, no longer requiring the pipeline of specialized systems used in statistical machine learning.

Unlike the traditional phrase-based translation system which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

Human and Machine Translation:

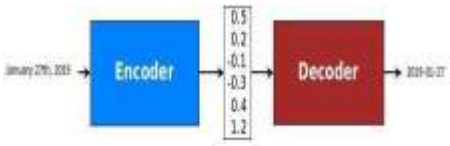


Fig 1 Human and Machine Translation

C. ENCODER

The task of the encoder is to provide a representation of the input sentence. The input sentence is a sequence of words (human readable dates with maximum 30-character length), for which I first consult the embedding matrix using one hot encoding. Then, as in the basic language model described previously, I process these words with a recurrent neural network using Long short term memory (LSTM). This results in hidden states that encode each word with its left context, i.e., all the preceding words. To also get the right context, I

also build a recurrent neural network that runs right-to-left, or more precisely, from the end of the sentence to the beginning. Having two recurrent neural networks running in two directions is called a bi-directional recurrent neural network.

D. DECODER

The decoder is a recurrent neural network. It takes some representation of the input context (more on that in the next section on the attention mechanism) and the previous hidden state and output word prediction and generates a new hidden decoder state and a new output word prediction.

If I use LSTMs for the encoder, then I also use LSTMs for the decoder. From the hidden state. I now predict the output word. This prediction takes the form of a probability distribution over the entire output vocabulary. If I have a vocabulary of, say, 50,000 words, then the prediction is a 50,000-dimensional vector, each element corresponding to the probability predicted for one word in the vocabulary.

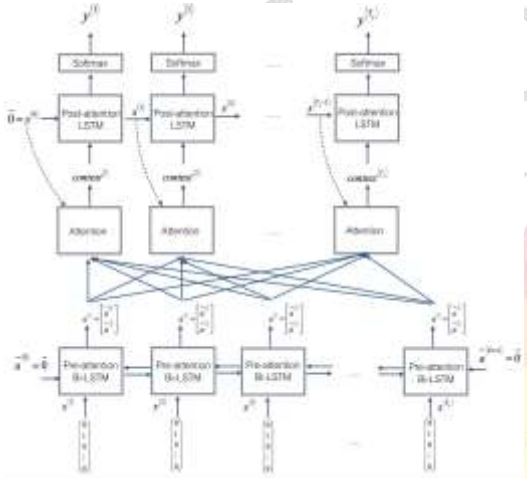


Fig 2 Attention Mechanism

E. Attention Mechanism

I currently have two loose ends. The decoder gave us a sequence of word representations hj = (←− hj , −→hj) and the decoder expects a context ci at each step i. I now describe the attention mechanism that ties these ends together. The attention mechanism is hard to visualize using our typical neural network graphs. The attention mechanism is informed by all input word representations (←− hj, −→hj) and the previous hidden state of the decoder si−1, and it produces a context state ci . The motivation is that I want to compute an association between the decoder state and each input word. Based on how strong this association is, or in other words how relevant each input word is to produce the next output word, I want to weight the impact of its word representation.

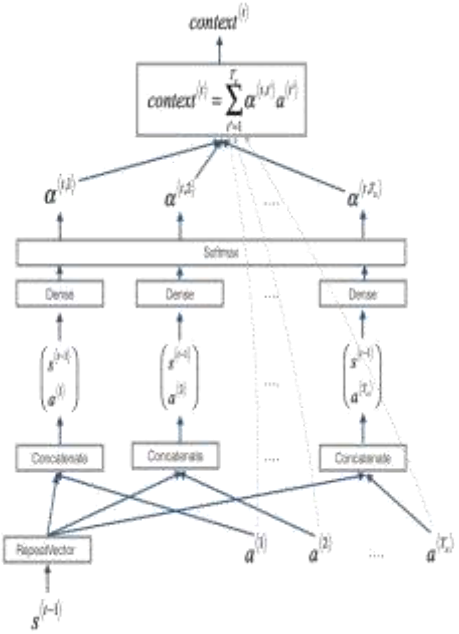


Fig 3 Training Model

F. Trainging the model

With the complete model in hand, I can now take a closer look at training. One challenge is that the number of steps in the decoder and the number of steps in the encoder varies with each training example. Sentence pairs consist of sentences of different length, so I cannot have the same computation graph for each training example but instead have to dynamically create the computation graph for each of them. This technique is called unrolling the recurrent neural networks.

Practical training of neural machine translation models requires GPUs which are well suited to the high degree of parallelism inherent in these deep learning models (just think of the many matrix multiplications). To increase parallelism even more, I process several sentence pairs (say, 100) at once. This implies that I increase the dimensionality of all the state tensors. To give an example, I represent each input word in specific sentence pair with a vector hj . Since I already have a sequence of input words, these are lined up in a matrix. When I process a batch of sentence pairs, I again line up these matrices into a 3-dimensional tensor. Similarly, to give another example, the decoder hidden state Si is a vector for each output word. Since I process a batch of sentences, I line up their hidden states into a matrix. Note that in this case it is not helpful to line up the states for all the output words, since the states are computed sequentially.

To summarize, training consists of the following steps:

- Shuffle the training corpus (to avoid undue biases due to temporal or topical order)
- Break up the corpus into maxi-batches
- Break up each maxi-batch into mini-batches
- Process each mini-batch, gather gradients

- Apply all gradients for a maxi-batch to update the parameters.

Typically, training neural machine translation models takes about 5–15 epochs (passes through entire training corpus). A common stopping criterion is to check progress of the model on a validation set (that is not part of the training data) and halt when the error on the validation set does not improve. Training longer would not lead to any further improvements and may even degrade performance due to overfitting.

III. MODEL CREATION

There are two separate LSTMs in this model, pre-attention Bi-LSTM, post-attention LSTM. The pre-attention Bi-LSTM goes through TxTx time steps, the post-attention LSTM goes through TyTy time steps.

The LSTM has both the output activation s⟨t⟩s⟨t⟩ and the hidden cell state c⟨t⟩c⟨t⟩. The post-activation LSTM at time tt does will not take the specific generated y⟨t−1⟩y⟨t−1⟩ as input, it only takes s⟨t⟩s⟨t⟩ and c⟨t⟩c⟨t⟩ as input. I have designed the model this way, because there isnot as strong a dependency between the previous character and the next character in a YYYY-MM-DD date.

I use a⟨t⟩=[a→⟨t⟩;a←⟨t⟩]a⟨t⟩=[a→⟨t⟩;a←⟨t⟩] to represent the concatenation of the activations of both the forward-direction and backward-directions of the pre-attention Bi-LSTM.

The architecture has a RepeatVector node to copy s⟨t−1⟩s⟨t−1⟩'s value TxTx times, and then Concatenation to concatenate s⟨t−1⟩s⟨t−1⟩ and a⟨t⟩a⟨t⟩ to compute e⟨t,t′e⟨t,t ′, which is then passed through a softmax to compute α⟨t,t ′⟩α⟨t,t′⟩.

Implementation functions

one_step_attention(): At step tt, given all the hidden states of the Bi-LSTM

([a<1>,a<2>,...,a<Tx>][a<1>,a<2>,...,a<Tx>]) and the previous hidden state of the second LSTM (s<t−1>s<t−1>)

one_step_attention() will compute the attention weights ([α<t,1>,α<t,2>,...,α<t,Tx>][α<t,1>,α<t,2>,...,α<t,Tx>]) and output the context vector.

$$context^{<t>} = \sum_{t'=0}^{T_x} \alpha^{<t,t'>} a^{<t'>}$$

Model

First phase runs the input through a Bi-LSTM to get back

[a<1>,a<2>,...,a<Tx>][a<1>,a<2>,...,a<Tx>].

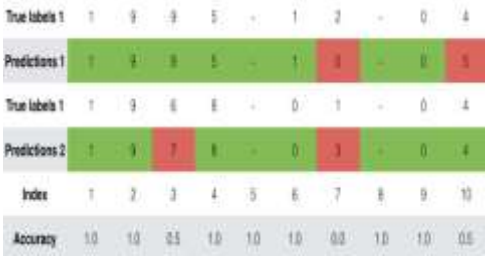
Then, it calls one_step_attention() TyTy times (for loop). At each iteration of this loop, it gives the computed context vector c<t>c<t> to the second LSTM and runs the output of the LSTM through a dense layer with softmax activation to generate a prediction ŷ <t>y^<t>.

Neural Machine Translation (NMT) is a new approach to machine translation that has shown promising results that are comparable to traditional approaches.

IV. PERFORMANCE EVALUATION

Defined model is verified using categorical_crossentropy loss, a custom Adam optimizer (learning rate = 0.005, β1=0.9β1=0.9, β2=0.999β2=0.999, decay = 0.01) and ['accuracy'] metrics, while training, observing the loss as well as the accuracy on each of the 10 positions of the output.

Table 1 loss and accuracy



Thus, dense_2_acc_8: 0.89 means that predicting the 7th character of the output correctly 89% of the time in the current batch of data.

Executed the model for longer and saved the weights. Run the next cell to load our weights.

V. CONCLUSION

Neural Machine translation models can be used to map from one sequence to another. They are useful not just for translating human languages (like Hindi->English) but also for tasks like date format translation.

An attention mechanism allows a network to focus on the most relevant parts of the input when producing a specific part of the output.

A network using an attention mechanism can translate from inputs of length TxTx to outputs of length TyTy, where TxTx and TyTy can be different.

REFERENCES

[1] Yoonjung Choi, Youngho Kim, Sung-Hyon Myaeng Domain-specific sentiment analysis using contextual feature generation(2009).

[2] Mikalai Tsytsarau,University of Trento, Trento, Italy.Sihem Amer-Yahia, Laboratoire d'Informatique de Grenoble, Grenoble, France. Themis Palpanas, University of Trento, Trento, Italy. J. Clerk Efficient sentiment correlation for large-scale demographics (2013).

[3] Kaiming Li Northwestern Polytechnical Univerisity, China and The University of Georgia. Human-centered attention models for video summarization (2010)

[4] Ming-Yu Wang, Xing Xie, Wei-Ying Ma, Hong-Jiang ZhangMicrosoft Research Asia, Beijing, P.R.China R. MobiPicture: browsing pictures on mobile devices

[5] A Statistical Approach to Machine Translation, 1990.

[6] Review Article: Example-based Machine Translation, 1999.

[7] Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, 2014.

[8] Neural Machine Translation by Jointly Learning to Align and Translate, 2014.

[9] Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016.

[10] Sequence to sequence learning with neural networks, 2014.

[11] Recurrent Continuous Translation Models, 2013.
[12] Continuous space translation models for phrase-based statistical machine translation, 2013.
[13] Jürgen Schmidhuber, Sepp Hochreiter Long Short-Term Memory (1997)